TU Clausthal

Clausthal University of Technology

Local Matrix Adaptation in Topographic Neural Maps

Banchar Arnonkijpanich [1], **Barbara Hammer** [2], **Alexander Hasenfuss** [2]

IfI Technical Report Series **IFI-08-07**

Impressum

Publisher: Institut für Informatik, Technische Universität Clausthal
Julius-Albert Str. 4, 38678 Clausthal-Zellerfeld, Germany
Editor of the series: Jürgen Dix
Technical editor: Wojciech Jamroga
Contact: wjamroga@in.tu-clausthal.de
URL: http://www.in.tu-clausthal.de/forschung/technical-reports/
ISSN: 1860-8477

The IfI Review Board

Local Matrix Adaptation in Topographic Neural Maps

Banchar Arnonkijpanich [1], Barbara Hammer [2], Alexander Hasenfuss [2]

1 – Khon Kaen University, Faculty of Science,
Department of Mathematics, Thailand
2 – Clausthal University of Technology,
Department of Computer Science, Germany

Abstract

The self-organizing map (SOM) and neural gas (NG) and generalizations thereof such as the generative topographic map constitute popular algorithms to represent data by means of prototypes arranged on a (hopefully) topology representing map. However, most standard methods rely on the Euclidean metric, hence the resulting clusters are isotropic and they cannot account for local distorsions or correlations of data. In this contribution, we extend prototype-based clustering algorithms such as NG and SOM towards a more general metric which is given by a full adaptive matrix such that ellipsoidal clusters are accounted for. Thereby, the approach relies on a natural extension of the standard cost functions of NG and SOM (in the form of Heskes) and is conceptually intuitive. We derive batch optimization learning rules for prototype and matrix adaptation based on these generalized cost functions and we show convergence of the algorithm. Thereby, it can be seen that matrix learning implicitly performs local principal component analysis (PCA) and the local eigenvectors correspond to the main axes of the ellipsoidal clusters. Thus, the proposal also provides a cost function associated to alternative proposals in the literature which combine SOM or NG with local PCA models. We demonstrate the behavior of the proposed model in several benchmark examples and in an application to image compression.

1 Introduction

The self-organizing map (SOM) as proposed by Kohonen constitutes one of the most popular data inspection and visualization tools due to its intuitive, robust, and flexible behavior with numerous applications ranging from web and text mining up to telecommunications and robotics [14]. Data are represented by means of typical prototypes which are arranged on a fixed lattice structure, often a low-dimensional regular lattice such that visualization of data is easily possible. In this respect, the SOM generalizes standard vector quantization by integrating a priorly fixed prototype topology, this way achieving a much more robust method which also extracts topological information about the given data. Neural gas (NG) as introduced by Martinetz transfers ideas

of SOM towards a data optimum neighborhood structure such that a representation by prototypes and topographic mapping of data can be achieved without any constraints on the topology given by a prior lattice structure [19]. Both methods have in common that they extract typical prototypes from the data which are arranged according to the local data characteristics. Thereby, neighborhood integration accounts for a very robust behavior of the models. A variety of alternative prototype based clustering methods have been proposed including standard vector quantization e.g. by means of the popular k-means algorithm, probabilistic extensions such as mixture of Gaussians, or fuzzy variants of prototype-based clustering [3, 10, 28].

Standard NG and SOM are based on the Euclidean metric. Correspondingly, the found clusters are isotropic with spherical class boundaries. Several methods extend SOM and NG towards more general distance measures, such as median or relational clustering which allows to process general dissimilarity data given by pairwise distances only [6, 9]. These methods, however, use a fixed priorly chosen metric. For the Euclidean setting, a number of approaches which adapt the distance calculation to the data at hand have been proposed such that more general cluster shapes can be accounted for.

The methods [23, 24] extend the setting towards a semi-supervised scenario and they adapt the metric such that the aspects which are relevant for the supervised labeling of data are emphasized in the visualization. Being semi-supervised, these settings require additional knowledge for the adaptation of the metric parameters.

Methods which are solely based on the unsupervised training data for the neural map include the popular adaptive subspace SOM which extracts invariant features in terms of invariant local subspaces attached to the prototypes [15]. A similar idea is proposed in the approaches [17, 21, 26, 1, 13] where prototypes are enriched by vectors corresponding to the local main principal directions of the data. These methods combine principal component analysis (PCA) techniques such as Oja's rule and extensions thereof with a vector quantization scheme such as SOM, possibly using an enriched metric. Thereby, the approaches rely on heuristics, or the learning rules are derived separately for clustering and PCA, respectively. Partially, the methods also formally establish stability of parts of the learning rules such as [21]. Up to our knowledge, however, none of these approaches derives the learning rules of vector quantization and determination of the principal directions from a uniform cost function. One exception is the approach presented in [27] which derives learning rules of class centers and principle directions from a uniform statistical approach which models PCA by means of a latent space model. Learning rules can be derived by means of the EM algorithm – however, the simplicity of most other models is not preserved and the method is sensitive to initialization of the parameters. We will argue in this approach that local PCA learning and vector quantization can be derived from a cost function which constitutes a simple extension of the standard cost function of NG towards an adaptive metric. As a side effect, convergence of a robust batch optimization scheme can be easily shown.

There exist several approaches in the literature which enrich the metric used for vector quantization towards a general adaptive form. One very elegant possibility represents the metric by a full matrix which can take an adaptive weighting of the dimensions

as well as correlations of dimensions into account. This scheme has been integrated into standard k-means algorithm and fuzzy extensions thereof as well as supervised prototype-based learning schemes such as learning vector quantization [22, 12, 25]. Interestingly, for k-means and fuzzy-k-means, matrix adaptation corresponds to an estimation of the local Mahalanobis distance of data. Hence the local principal directions are implicitly determined in these approaches and clusters are given by ellipsoids aligned along the local principal directions of data. Up to our knowledge, no such extensions of neural vector quantization schemes such as NG and SOM towards a general matrix exist.

In this approach, we extend NG and SOM towards a general adaptive matrix which characterizes the metric. We derive update rules of the parameters based on a uniform underlying cost function of NG and SOM in the variant as proposed by Heskes [11], relying on batch optimization schemes. The resulting update rules for the matrix correspond to a generalized Mahalanobis distance, hence the method can be linked to local PCA methods. We show convergence of the update rules, and we demonstrate the behavior in a variety of benchmark examples and an application to image compression. It can be seen that a very robust method results which is very insensitive to initialization of the parameters.

2 Topographic neural maps

Assume data points $\vec{x} \in \mathbb{R}^m$ are distributed according to a probability distribution P. The goal of prototype based clustering is to represent data by means of prototype vectors $\vec{w}^i \in \mathbb{R}^m$, $i = 1, \ldots, n$, such that they represent the distribution as accurately as possible. Thereby, a data point $\vec{x} \in \mathbb{R}^m$ is represented by the winning prototype $\vec{w}^{I(\vec{x})}$ which is the prototype located closest to the data point, i.e.

$$I(\vec{x}) = \mathrm{argmin}_i \{d(\vec{x}, \vec{w}^i)\} \tag{1}$$

measured according to the squared Euclidean distance

$$d(\vec{x}, \vec{w}^i) = (\vec{x} - \vec{w}^i)^t (\vec{x} - \vec{w}^i)\,. \tag{2}$$

The mathematical objective of vector quantization is to minimize the quantization error

$$E_{\mathrm{VQ}}(\vec{w}) \sim \frac{1}{2} \sum_{i=1}^{n} \int \delta_{i,I(\vec{x})} \cdot d(\vec{x}, \vec{w}^i)\, P(d\vec{x}) \tag{3}$$

where $\delta_{i,j}$ denotes the Kronecker delta symbol. Given a finite set of training examples $\vec{x}^1, \ldots, \vec{x}^p$, the popular k-means algorithm optimizes the corresponding discrete cost function $1/2 \cdot \sum_{i=1}^{n} \sum_{j=1}^{p} \delta_{i,I(\vec{x}^j)} \cdot d(\vec{x}^j, \vec{w}^i)$ by means of an EM scheme, subsequently optimizing data assignments and prototype locations by means of the updates

$$k_{ij} := \delta_{i,I(\vec{x}^j)}, \quad \vec{w}^i := \sum_j k_{ij} \vec{x}^j \Big/ \sum_j k_{ij} \tag{4}$$

Since the cost function (3) is usually highly multimodal, k-means clustering gets easily stuck in local optima and multiple restarts are necessary to achieve a good solution.

NG and SOM offer alternatives which integrate neighborhood cooperation of the prototypes and, this way, achieve robustness of the results with respect to prototype initialization. In addition, neighborhood cooperation accounts for a topological arrangement of prototypes such that browsing and, in the case of SOM with low-dimensional lattice structure, direct visualization of data become possible. The cost function of NG is given by

$$E_{\mathrm{NG}}(\vec{w}) \sim \frac{1}{2} \sum_{i=1}^{n} \int h_\sigma(k_i(\vec{x})) \cdot d(\vec{x}, \vec{w}^i)\, P(d\vec{x}) \tag{5}$$

where $k_i(\vec{x}) \in \{0, \ldots, n-1\}$ constitutes a permutation of prototypes arranged according to the distance for every $\vec{x}$. If the distances are mutually disjoint, it is given by

$$k_i(\vec{x}) = |\{\vec{w}^j \mid d(\vec{x}, \vec{w}^j) < d(\vec{x}, \vec{w}^i)\}| \tag{6}$$

If distances coincide, ties are broken arbitrarily. $h_\sigma(t) = \exp(-t/\sigma)$ is a Gaussian shaped curve with neighborhood range $\sigma > 0$. Obviously, for vanishing neighborhood $\sigma \to 0$, the quantization error (3) is recovered. NG is usually optimized by means of a stochastic gradient descent method. Alternatively, for a given finite data set as above, the corresponding discrete cost function $1/2 \cdot \sum_{i=1}^{n} \sum_{j=1}^{p} h_\sigma(k_i(\vec{x}^j)) \cdot d(\vec{x}^j, \vec{w}^i)$ can be optimized in a batch scheme in analogy to k-means (4) using the update rules

$$k_{ij} := k_i(\vec{x}^j), \quad \vec{w}^i := \sum_j h_\sigma(k_{ij}) \vec{x}^j \Big/ \sum_j h_\sigma(k_{ij}) \tag{7}$$

as pointed out in the approach [6]. During training, the neighborhood range σ is annealed to 0 such that the quantization error is recovered in final steps. In intermediate steps, a neighborhood structure of the prototypes is determined by the ranks according to the given training data. This choice, on the one hand, accounts for a high robustness of the algorithm with respect to local minima of the quantization error, on the other hand, NG can be extended to extract an optimum topological structure from the data by means of Hebbian updates [20]. Due to its simple adaptation rule, the independence of a prior lattice, and the independence of initialization because of the integrated neighborhood cooperation, NG offers a simple and highly effective algorithm for data clustering.

SOM uses the adaptation strength $h_\sigma(nd(I(\vec{x}^j), i))$ instead of $h_\sigma(k_i(\vec{x}^j))$, where nd is a priorly chosen, often two-dimensional neighborhood structure of the neurons. A low-dimensional lattice offers the possibility to visualize data easily. However, if the primary goal is clustering, a fixed topology puts restrictions on the map and topology preservation can be violated. The original SOM does not possess a cost function in the continuous case and its mathematical investigation is difficult [4]. A slight change of the winner notation can be associated to a cost function as pointed out by Heskes [11].

We substitute the winner by

$$I^*(\vec{x}) = \mathrm{argmin}_i \sum_{l=1}^{n} h_\sigma(nd(i,l)) d(\vec{x}, \vec{w}^l)\,, \tag{8}$$

i.e. the prototype which is closest to $\vec{x}$ averaged over the local neighborhood of the prototype. This way, SOM optimizes the cost

$$E_{\mathrm{SOM}}(\vec{w}) \sim \frac{1}{2} \sum_{i=1}^{n} \int \delta_{i,I^*(\vec{x})} \cdot \sum_{l=1}^{n} h_\sigma(nd(i,l)) \cdot d(\vec{x}, \vec{w}^l)\, P(d\vec{x}) \tag{9}$$

as pointed out by Heskes [11]. Like NG, SOM is usually optimized by means of a stochastic gradient descent. For a given finite set of training data as beforehand, alternative batch optimization is possible characterized by the iterative updates

$$k_{ij} := \delta_{i,I^*(\vec{x}^j)}, \quad \vec{w}^i := \sum_{j,l} k_{lj} h_\sigma(nd(l,i)) \vec{x}^j \Big/ \sum_{j,l} k_{lj} h_\sigma(nd(l,i)) \tag{10}$$

of assignments and prototype vectors. The neighborhood strength is thereby annealed $\sigma \to 0$ (or a small nonvanishing value, respectively, depending on the application area) during training. This yields to a very fast adaptation towards a topology preserving map (for nonvanishing σ) such that browsing and, in the case of a low dimensional lattice, data visualization become possible. Topological mismatches can occur due to two reasons, a mismatch of data and lattice topology or a too fast adaptation, as discussed e.g. in [8].

It has been proved in [6] that batch SOM and batch NG converge to a local optimum of the corresponding cost functions in a finite number of steps. Both methods offer clustering schemes which require only a few parameters: the number of clusters and the number of training epochs, such that robust methods are obtained which do not need a time consuming optimization of additional meta-parameters such as the learning rate.

3 Matrix learning

Classical NG and SOM rely on the Euclidean metric which induces isotropic cluster shapes. Thus they cannot account for a different scaling or correlation of the data dimensions. General ellipsoidal shapes can be achieved by the generalized metric form

$$d_{\Lambda_i}(\vec{x}, \vec{w}^i) = (\vec{x} - \vec{w}^i)^t \Lambda_i (\vec{x} - \vec{w}^i) \tag{11}$$

instead of the squared Euclidean metric (2) where $\Lambda_i \in \mathbb{R}^{m \times m}$ is a symmetric positive definite matrix with $\det \Lambda_i = 1$. These constraints are necessary to guarantee that the resulting formula defines a metric which does not degenerate to a trivial form ($\Lambda_i = 0$ constituting an obvious trivial optimum of the cost functions). A general matrix can

account for correlations and appropriate nonuniform scaling of the data dimensions. Obviously, an optimum matrix Λ_i is not known before training. Therefore we optimize the parameter Λ_i according to the given training data. The following cost functions result:

$$E_{\mathrm{NG}}(\vec{w}, \Lambda) \sim \frac{1}{2} \sum_{i=1}^{n} \int h_\sigma(k_i(\vec{x})) \cdot d_{\Lambda_i}(\vec{x}, \vec{w}^i)\, P(d\vec{x}) \tag{12}$$

for matrix NG and

$$E_{\mathrm{SOM}}(\vec{w}, \Lambda) \sim \frac{1}{2} \sum_{i=1}^{n} \int \delta_{i, I^*(\vec{x})} \cdot \sum_{l=1}^{n} h_\sigma(nd(i,l)) \cdot d_{\Lambda_i}(\vec{x}, \vec{w}^l)\, P(d\vec{x}) \tag{13}$$

for matrix SOM, whereby the assignments $k_i(\vec{x})$ and $I^*(\vec{x})$, respectively, are computed based on d_{Λ_i}, and Λ_i is restricted to symmetric positive definite forms with $\det \Lambda_i = 1$.

Matrix NG

We derive batch optimization schemes for matrix NG and matrix SOM based on these cost functions. First, we consider matrix NG. We assume that a finite number of training data $\vec{x}^1, \ldots, \vec{x}^p$ are given and we consider the associated discrete cost function of NG. As for batch NG, we introduce hidden variables k_{ij} for $i = 1, \ldots, n$, $j = 1, \ldots, p$ which constitute a permutation of $\{0, \ldots, n-1\}$ for every fixed j, and we extend the cost function to

$$E_{\mathrm{NG}}(\vec{w}, \Lambda, k_{ij}) = \sum_{ij} h_\sigma(k_{ij}) \cdot d_{\Lambda_i}(\vec{x}^j, \vec{w}^i)$$

where the hidden variables k_{ij} take the place of the original values $k_i(\vec{x}^j)$, the latter depending on $\vec{w}$ and Λ. Obviously, optimum assignments k_{ij} fulfill the equality $k_{ij} = k_i(\vec{x}^j)$ such that $E_{\mathrm{NG}}(\vec{w}, \Lambda, k_{ij}) = E_{\mathrm{NG}}(\vec{w}, \Lambda)$ for optimum choices of the hidden variables k_{ij}. Batch optimization, in turn, optimizes the hidden variables k_{ij} for fixed Λ and $\vec{w}$, and it determines optimum parameters Λ and $\vec{w}$ given fixed assignments k_{ij}. Because the cost function $E_{\mathrm{NG}}(\vec{w}, \Lambda, k_{ij})$ has a much simpler form w.r.t. $\vec{w}$ and Λ than the original one, optima can be determined analytically.

The learning algorithm is obtained by computing the optima of the function $E_{\mathrm{NG}}(\vec{w}, \Lambda, k_{ij})$ for fixed k_{ij} and, in turn, for fixed $\vec{w}^i$ and Λ_i, respectively.

- Optimization w.r.t k_{ij}: As already stated, optimum assignments obviously fulfill the equation
$$k_{ij} = k_i(\vec{x}^j) \tag{14}$$
given the constraints on k_{ij} and fixed $\vec{w}$ and Λ.

- Optimization w.r.t. $\vec{w}^i$: Given fixed k_{ij}, the directional derivative of E_{NG} with respect to $\vec{w}^i$ into direction $\xi \in \mathbb{R}^m$ has the form
$$\sum_j h_\sigma(k_{ij})(\Lambda_i(\vec{w}^i - \vec{x}^j))^t \xi \stackrel{!}{=} 0.$$

For a potential local optimum, this must be 0 for all directions ξ. Therefore, we find $\sum_j h_\sigma(k_{ij})(\vec{w}^i - \vec{x}^j) = 0$ for all i and, hence

$$\vec{w}^i = \frac{\sum_j h_\sigma(k_{ij})\vec{x}^j}{\sum_j h_\sigma(k_{ij})} \tag{15}$$

Note that the function E_{NG} constitutes a quadratic form with respect to $\vec{w}^i$ with diagonal Hessian matrix with entries $\sum_j h_\sigma(k_{ij})\Lambda_i$, which is positive definite since the matrices Λ_i are. Thus, the function must have a local minimum which is also the global minimum. Thus, the above value constitutes a global optimum for $\vec{w}^i$ given fixed k_{ij} and matrices Λ_i.

- Optimization w.r.t. Λ_i: We ignore the constraint of symmetry and positive definiteness of Λ_i for the moment. Taking into account the constraint $\det \Lambda_i = 1$, we arrive at the Lagrange function

$$L = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{p} h_\sigma(k_{ij}) \cdot d_{\Lambda_i}(\vec{x}^j, \vec{w}^i) - \sum_{i=1}^{n} \lambda_i(\det \Lambda_i - 1)$$

with Lagrange parameters $\lambda_i \in \mathbb{R}$. Taking the derivative w.r.t Λ_i yields $\sum_j h_\sigma(k_{ij})(\vec{x}^j - \vec{w}^i)(\vec{x}^j - \vec{w}^i)^t - \lambda_i(\det \Lambda_i \cdot \Lambda_i^{-1}) \stackrel{!}{=} 0$. This must be 0 for local optima within the feasable range of the parameters, hence

$$\Lambda_i = \left(\sum_j h_\sigma(k_{ij})(\vec{x}^j - \vec{w}^i)(\vec{x}^j - \vec{w}^i)^t\right)^{-1} \lambda_i$$

because $\det \Lambda_i = 1$. We set

$$S_i := \sum_j h_\sigma(k_{ij})(\vec{x}^j - \vec{w}^i)(\vec{x}^j - \vec{w}^i)^t \tag{16}$$

Using this definition, we have $S_i\Lambda_i = \lambda_i I$, I being the identity, and, hence, $\det(S_i\Lambda_i) = \det S_i = \lambda_i^n$, thus $\lambda_i = (\det S_i)^{1/n}$. Thus,

$$\Lambda_i = S_i^{-1}(\det S_i)^{1/n} \tag{17}$$

This is well defined if S_i is invertible, which is the case if there exist at least n linearly independent vectors $\vec{x}^j - \vec{w}^i$ in the data set. Otherwise, the pseudoinverse should be used. Note that Λ_i is symmetric because S_i is symmetric, and Λ_i is positive definite if S_i is positive definite, which is also guaranteed if there exist n linearly independent vectors $\vec{x}^j - \vec{w}^i$. Thus the resulting matrix fulfills the conditions of a metric.

We want to investigate whether this choice of the matrices gives a global optimum of the cost function for fixed assignments k_{ij}, or whether this choise only

gives a local optimum or a saddle point. We assume that S_i is invertible. We make the even stronger assumption that the variance of the data does not vanish in any dimension. Note that we can represent every symmetric matrix Λ_i with determinant 1 uniquely in the form $\Lambda_i = U_i^t D_i U_i$ where U_i is contained in the orthogonal group $O(n)$ and D_i is a diagonal matrix with determinant 1. Hence, alternatively to Lagrange optimization, a direct optimization with respect to matrices of this form could be done and it would lead to at most the same possible local optimum. Note that matrix multiplication is continuous, as is the cost function E_{NG} with respect to the matrices Λ_i. $O(n)$ is a compact Lie group. Hence we arrive at the borders of the feasable range of Λ_i by letting a diagonal element d_{ij} of D_i tend to infitity: i.e. at the borders, $d_{ij} \to \infty$ for at least one diagonal element d_{ij} of at least one matrix D_i. The resulting value of the cost function is $\sum_{ij} h_\sigma(k_{ij}) \cdot (\vec{x}^j - \vec{w}^i)^t \cdot U_i^t \cdot D_i \cdot U_i \cdot (\vec{x}^j - \vec{w}^i) = \sum_{ijl} h_\sigma(k_{ij}) \cdot |\mathrm{proj}_l^i(\vec{x}^j - \vec{w}^i)|^2 \cdot d_{il}$ where $\mathrm{proj}_l^i(\vec{x})$ describes the coefficient l of the projection of vector $\vec{x}$ to the orthonormal system given by U_i. Note that, since we assume nonvanishing variance of the data in every direction, the inequality $|\mathrm{proj}_l^i(\vec{x}^j - \vec{w}^i)|^2 > 0$ holds for every i, j, and l. Thus $d_{ij} \to \infty$ causes $E_{\mathrm{NG}} \to \infty$. Hence, the cost function increases beyond limits for the borders of the feasable range of the matrices. Therefore, every optimum of the cost function with respect to λ_i for fixed prototypes and assignments must be a local optimum, and at least one such optimum exists. Since there is only one potential candidate for this position, we have found a global optimum of the cost function given fixed prototypes and assignments.

- Assume matrices Λ_i and prototypes $\vec{w}^i$ are varied at the same time. Also in this case, a global optimum is found, given fixed assignments k_{ij} as can be seen as follows: We reach the borders of the feasable range if either a prototype component $\vec{w}^i$ reaches $\pm\infty$ or a matrix element reaches $d_{ij} \to \infty$. As before, the cost function becomes $\sum_{ijl} h_\sigma(k_{ij}) \cdot |\mathrm{proj}_l^i(\vec{x}^j - \vec{w}^i)|^2 \cdot d_{il}$. Since the variance is nonvanishing in every direction, there exists for every position $\vec{w}^i$ of the space and every orthogonal projection a data point j such that $|\mathrm{proj}_l^i(\vec{x}^j - \vec{w}^i)| > \epsilon$ for some $\epsilon > 0$. Hence, if some component d_{il} goes to infinity, the cost function increases beyond bounds. If a prototype component tends to infinity, either the term $|\mathrm{proj}_l^i(\vec{x}^j - \vec{w}^i)|^2 \cdot d_{il}$ also tends to infinity for at least one l, since the projection is orthogonal, or the term d_{il} vanishes. In the latter case, because of $\det \Lambda_i = 1$, another component $d_{il'}$ tends to infinity, thus, in both cases, the cost function increases beyond bounds. Thus, also if prototypes and matrices are optimized at the same time, one global optimum must exist, which is given by the above formulas.

While we do not need the proof that the respective values constitute global optima for the derivation of the algorithm itself, we will use this fact when addessing convergence of the algorithm. Using these equations, batch optimization for matrix NG is given by the following procedure:

init $\vec{w}^i$ (e.g. using random data points)
init Λ_i as identity I
repeat until convergence
determine k_{ij} using (14)
determine $\vec{w}^i$ using (15)
determine Λ_i using (17)

The matrix S_i corresponds to the correlation of the data centered at prototype $\vec{w}^i$ and weighted according to its distance from the prototype. For vanishing neighborhood $\sigma \to 0$, the standard correlation matrix of the receptive field is obtained and the distance corresponds to the Mahalanobis distance in the receptive field of the prototype. The Mahalanobis distance corresponds to a scaling of the principal axes of the data space by the inverse eigenvalues in the eigendirections. Thus, ellipsoidal cluster shapes arise which main directions are centered along the local principal directions of the data, and the scaling is elongated along the main principal components - in these directions, the eigenvalues are large, i.e. its inverse is small; therefore, deviations along these main directions are better tolerated by the induced metric than deviations along the minor components. We conclude that matrix learning performs implicit local PCA and the main local principal components at $\vec{w}^i$ can be discovered by looking at the minor principal components of Λ_i. Unlike standard PCA, neighborhood cooperation is applied to both, prototype adaptation and matrix learning during batch training, such that a regularization of matrix learning is given due to the neighborhood which is beneficial e.g. for small clusters.

We would like to point out that it is easily possible to use only one global matrix Λ instead of local matrices Λ_i attached to the prototypes, as proposed in [22]. The resulting formula for Λ is easily obtained from the ones given above by means of the sum of the local matrices S_i. This way, the number of parameters is reduced and a global Mahalanobis distance which is derived from the data centered around their respective class prototypes is obtained.

Matrix SOM

Matrix learning for SOM can be derived in a similar way: We introduce hidden variables k_{ij} for $i = 1, \ldots, n$, $j = 1, \ldots, p$ which are contained in $\{0, 1\}$ such that $\sum_i k_{ij} = 1$ for all $\vec{x}^j$. The cost function of SOM for a given discrete data set is substituted by

$$E_{\mathrm{SOM}}(\vec{w}, \Lambda, k_{ij}) = \frac{1}{2} \sum_{ij} k_{ij} \cdot \sum_l h_\sigma(nd(i,l)) \cdot d_{\Lambda_i}(\vec{x}^j, \vec{w}^l)$$

where the hidden variables k_{ij} substitute the term $\delta_{i,I^*(\vec{x}^j)}$ which depends on $\vec{w}$ and $\vec{\Lambda}$. For optimum assignments k_{ij}, the two values are obviously identical. As before, iterative optimization of $E_{\mathrm{SOM}}(\vec{w}, \Lambda, k_{ij})$ w.r.t. $\vec{w}$, Λ and k_{ij} is done, using the following identities:

- Optimization w.r.t. k_{ij}: Optimum values are

$$k_{ij} = \delta_{i,I^*(\vec{x}^j)} \tag{18}$$

- Optimization w.r.t. $\vec{w}^i$: Given fixed k_{ij}, the directional derivative w.r.t. $\vec{w}^i$ into direction ξ vanishes for all directions, hence

$$\vec{w}^i = \sum_{jl} k_{lj} h_\sigma(nd(l,i))\vec{x}^j / \sum_{jl} k_{lj} h_\sigma(nd(l,i)) \tag{19}$$

- Optimization w.r.t Λ_i: Given fixed k_{ij} and ignoring for the moment symmetry and positive definiteness of Λ_i, we can consider the Lagrange function

$$L = \frac{1}{2}\sum_{ij} k_{ij} \cdot \sum_l h_\sigma(nd(i,l)) \cdot d_{\Lambda_i}(\vec{x}^j, \vec{w}^l) - \sum_i \lambda_i(\det \Lambda_i - 1)$$

with Lagrange parameters $\lambda_i \in \mathbb{R}$. Setting the derivative w.r.t Λ_i to 0 we obtain

$$\Lambda_i = \left(\sum_{lj} k_{lj} h_\sigma(nd(l,i))(\vec{x}^j - \vec{w}^i)(\vec{x}^j - \vec{w}^i)^t \right)^{-1} \lambda_i$$

We set

$$S_i := \sum_{lj} k_{lj} h_\sigma(nd(l,i))(\vec{x}^j - \vec{w}^i)(\vec{x}^j - \vec{w}^i)^t \tag{20}$$

and obtain

$$\Lambda_i = S_i^{-1}(\det S_i)^{1/n} \tag{21}$$

as before, whereby, now, the correlation matrix S_i is measured taking the lattice structure of SOM into account. As before, Λ_i is symmetric because S_i is symmetric, and Λ_i is positive definite if S_i is positive definite, which is guaranteed if there exist n linearly independent vectors $\vec{x}^j - \vec{w}^i$. Note that vanishing neighborhood $\sigma \to 0$ yields the standard correlation, i.e. ellipsoidal clusters aligned along the local principal components as for matrix NG. As before, one can see that these potential local optima constitute global optima of the cost function given fixed assignments.

Convergence

Obviously, matrix adaptation can be performed for simple k-means as well yielding the standard local Mahalanobis distance. Matrix k-means is recovered for vanishing neighborhood $\sigma \to 0$ for both, matrix NG and matrix SOM. Here we exemplarily show convergence of matrix learning for NG, the argumentation for SOM and k-means being

similar. We assume that the variance of the data does not vanish in any direction. Note that the cost function of NG can be written as

$$E(\vec{w}, \Lambda) = \sum_{i=1}^{n} \sum_{j=1}^{p} f_1(k_{ij}(\vec{w}, \Lambda)) \cdot f_2^{ij}(\vec{w}, \Lambda) \tag{22}$$

where $f_1(k_{ij}) = h_\sigma(k_i(\vec{x}^j))$ and $f_2^{ij} = d_{\Lambda_i}(\vec{x}^j, \vec{w}^i)$.

Batch optimization substitutes the dependent values $k_{ij}(\vec{w}, \Lambda)$ by new hidden parameters which are optimized under the constraint that k_{ij} constitute a permutation of $\{0, \ldots, n-1\}$ for every fixed i. For optimum values k_{ij} given fixed $\vec{w}$, Λ, the equality $k_{ij} = k_{ij}(\vec{w}, \Lambda)$ holds. Batch clustering in turn finds optimum values $\vec{w}$ and, afterwards, Λ, given fixed assignments k_{ij} and it determines optimum assignments k_{ij} given fixed parameters $\vec{w}$ and Λ, as we have already shown. We can assume that the respective optima are unique, which is obvious from the formulas for $\vec{w}$ and Λ, and which can be achieved for k_{ij} by breaking ties deterministically.

Consider the function

$$Q(\vec{w}, \Lambda, \vec{w}', \Lambda') = \sum_{i=1}^{n} \sum_{j=1}^{p} f_1(k_{ij}(\vec{w}, \Lambda)) \cdot f_2^{ij}(\vec{w}', \Lambda') \tag{23}$$

where $k_{ij}(\vec{w}, \Lambda)$ denotes the unique optimum assignment of k_{ij} given fixed $\vec{w}$ and Λ of the cost function. It holds $E(\vec{w}, \Lambda) = Q(\vec{w}, \Lambda, \vec{w}, \Lambda)$. Denote by $\vec{w}(t)$ and $\Lambda(t)$ the values derived in batch clustering in epoch t. Then we find

$$\begin{aligned} E(\vec{w}(t+1), \Lambda(t+1)) &= Q(\vec{w}(t+1), \Lambda(t+1), \vec{w}(t+1), \Lambda(t+1)) \\ &\leq Q(\vec{w}(t), \Lambda(t), \vec{w}(t+1), \Lambda(t+1)) \end{aligned}$$

because $k_{ij}(\vec{w}(t+1), \Lambda(t+1))$ are optimum assignments for given $\vec{w}(t+1)$ and $\Lambda(t+1)$. Further, we find

$$\begin{aligned} Q(\vec{w}(t), \Lambda(t), \vec{w}(t+1), \Lambda(t+1)) &\leq \\ Q(\vec{w}(t), \Lambda(t), \vec{w}(t+1), \Lambda(t)) &\leq \\ Q(\vec{w}(t), \Lambda(t), \vec{w}(t), \Lambda(t)) &= E(\vec{w}(t), \Lambda(t)) \end{aligned}$$

because $\vec{w}(t+1)$ is chosen as optimum value for the assignments $k_{ij}(\vec{w}(t), \Lambda(t))$ and fixed $\Lambda(t)$, and $\Lambda(t+1)$ is chosen optimum for the assignments. $k_{ij}(\vec{w}(t), \Lambda(t))$ and fixed $\vec{w}(t+1)$ Hence

$$E(\vec{w}(t+1), \Lambda(t+1)) \leq E(\vec{w}(t), \Lambda(t))$$

Thus, the cost function (23) is decreased in consecutive steps. Since the assignments k_{ij} stem from a finite set and the respective optimum values are unique, the algorithm converges in a finite number of steps towards a fixed point $\vec{w}^*$, Λ^* of the algorithm.

Assume that, for this fixed point, no data points $\vec{x}^j$ lie at the borders of receptive fields (the latter being a set of measure 0 since optimization takes place for a finite

set). Then, the underlying cost function (22) is continuous at $\vec{w}^*$, Λ^*, and, since the assignments k_{ij} are discrete, the function $k_{ij}(\vec{w}, \Lambda)$ is constant in a vicinity of $\vec{w}^*$, Λ^*. Hence $E(\cdot)$ and $Q(\vec{w}^*, \Lambda^*, \cdot)$ are identical in a neighborhood of $\vec{w}^*$, Λ^*, i.e. local optima of Q correspond to local optima of E. As shown beforehand, we arrive at a local optimum of this function for fixed assignments k_{ij} simultaneously for the matrices and prototypes. Hence, we arrive at a local optimum of E itself, if no data points lie at borders of receptive fields for the final solution and the variance of the data is nonvanishing in every direction.

Complexity

We derived batch formulas for a full matrix adaptation of SOM and NG. Unlike previous work on matrix adaptation or related local PCA methods such as [17, 21, 26, 22] the algorithm has been derived from a single cost function which directly extends the standard cost function of NG and SOM and convergence of the algorithm can be guaranteed. Note that, unlike the approaches [17, 21, 26] neighborhood cooperation is included into the matrix update or the corresponding PCA, respectively, to make sure that a global cost function is optimized by the methods. Once the connection of matrix learning and PCA is established by means of this general derivation, heuristics which obtain an optimum matrix by means of alternative PCA schemes such as variations of the Oja rule (e.g. [21]) are justified since they arrive at the same result for Λ as the above method. Note that, to guarantee convergence of the algorithm, it is sufficient to guarantee improvement of the cost function in every step, but a global optimum of the considered parameter need not necessarily be found. Thus convergence of alternative optimization schemes including online schemes such as gradient methods directly follows.

Batch clustering usually converges after only a very small number of epochs since the update rules can be interpreted as (fast) Newton method [3, 6]. The main complexity of matrix learning is due to the update of the matrix Λ which requires matrix inversion and a normalization by means of the determinant, i.e. steps with complexity $\mathcal{O}(m^3)$, m being the data dimensionality. While this can easily be done for low dimensional data sets, it becomes infeasible for high dimensionality. In addition, numerical instabilities can occur for higher dimensionalities and full matrix adaptation, since the variance can become very small in the minor principal components of the data. For this reason, it is advisable to use the pseudoinverse or even regularizations instead of exact matrix inversion.

The approach [21] proposes an intuitive alternative regularization scheme to reduce numerical instabilities and to obtain a lower complexity: it computes only $k \ll m$ major principal components and it scales all remaining dimensions uniformly. Note that the weighting should not be set to 0 to guarantee the metric property and locality of clusters, i.e. eigenvalues of matrices Λ_i exactly 0 should be avoided. In the approach [21] the following metric is suggested (up to normalization which is achieved in [21] by substracting the logarithm of the determinant of the matrix):

$$d_\Lambda(\vec{x}, \vec{w}) = (\vec{x} - \vec{w})^t \Lambda (\vec{x} - \vec{w}) + \frac{1}{\lambda^*}((\vec{x} - \vec{w})^t(\vec{x} - \vec{w}) - (\vec{x} - \vec{w})^t \Omega\Omega^t(\vec{x} - \vec{w}))$$

where $\Lambda = \Omega D \Omega^t$ is the singular value decomposition of the rank k matrix Λ given by the k major principal components of the correlation matrix Ω and the inverse of the corresponding eigenvalues D, and λ^* is an estimation of the variance in the remaining $m-k$ directions. Note that this metric corresponds to the choice

$$(\vec{x}-\vec{w})^t \begin{pmatrix} \Omega \\ \Omega^\perp \end{pmatrix} \cdot \begin{pmatrix} D & 0 \\ 0 & M_\lambda \end{pmatrix} \cdot \begin{pmatrix} \Omega \\ \Omega^\perp \end{pmatrix}^t (\vec{x}-\vec{w})$$

where $\Omega^\perp$ denotes an orthogonal projection to the space dual to Ω, and M_λ refers to the diagonal matrix with elements $(1/\lambda^*)^{1/(m-k)}$. This way, only k principal components need to be computed explicitely.

We provide a theoretical justification for this procedure under the assumption that the average λ^* is small in comparison to the explicit eigenvalues in Λ in the appendix.

4 Experiments

We test the algorithm for several illustrative two dimensional examples and benchmarks from the UCI repository [2] as well as an application for image compression. For all experiments, initial neighborhood ranges between $n/2$ (for NG) and $n/12$ (for SOM) are used and the neighborhood is multiplicatively annealed to 0 during training. Training takes place for 100 epochs. The average results over 10 repetitions are reported. We train standard k-means, neural gas, and SOM with two-dimensional rectangular neighborhood structure with and without full matrix adaptation.

Gaussian clusters

The first dataset consists of 4 two-dimensional ellipsoidal clusters as shown in Fig. 1. We train NG, SOM, and k-means with full matrix adaptation and four prototypes. The final results are the same for all methods: after about 30 epochs, convergence can be observed and the found clusters well resemble the data clusters. For every step, we depict the major principal component of the data correlation matrix of the receptive field and the minor principal component of the matrix assigned to the prototypes. As already mentioned, these directions should become identical during training and, thus, implicit local PCA is performed by matrix NG. One can observe that the directions deviate at the beginning of training due to the influence of the neighborhood which causes an averaging of matrices. At the end, perfect agreement can be observed.

Spirals

The spirals data set is depicted in Fig. 2. It is trained using 25 prototypes. As before, convergence can be observed after few steps. The prototypes are located on the surface and the cluster shapes are adapted to the spiral, showing elongated ellipses at the outer regions and almost circular behavior in the middle. For SOM, the same behavior can

be observed, whereas k-means suffers from local optima. Fig. 3 displays the superior result obtained by neighborhood integration for SOM and NG.

Checkerboard

In the checkerboard data set, data are arranged according to a checkerboard structure with almost circular and elongated clusters, respectively, as depicted in Fig. 4 (top and bottom, respectively). 25 prototypes are adapted using matrix NG, SOM, and k-means. Obviously, NG and SOM yield perfect and robust results, whereas k-means suffers from local optima.

UCI classification data

We consider three benchmark datasets from the UCI repository of machine learning [2]:

- Iris: The popular iris data set consists of 150 data points assigned to 3 classes. Data are numerical with 4 dimensions.
- The Wisconsin diagnostic breast cancer data set contains 569 data of 2 classes represented by 32 dimensional vectors.
- The ionosphere data contains 351 examples in 2 classes, data are given as 34 dimensional vectors.

Class labels are available for the data sets, i.e. we can assign a label c_i to every data point $\vec{x}^i$ in the data. We evaluate the methods by the ability to group the data into the priorly given classes. For this purpose, we consider two different evaluation measures:

- *Cluster coherence:* Denote by $c(\vec{x}^i)$ the cluster given by a learned clustering, i.e. the number of the winning prototype. Then, we use the evaluation measure

$$C_1 := \sum_{i=1}^{p} \sum_{j=i+1}^{p} \frac{\delta\left(\delta(c_i = c_j) - \delta(I(\vec{x}^i) = I(\vec{x}^j))\right)}{0.5 \cdot p(p-1)}$$

  where δ is the indicator function $\delta(0) = 1$ and $\delta(t) = 0$ for $t \neq 0$, and the terms $c_i = c_j$ and $I(\vec{x}^i) = I(\vec{x}^j)$ refer to the question whether the labels c_i and c_j or the winner prototypes of $\vec{x}^i$ and $\vec{x}^j$, respectively, are identical. Obviously, C_1 measures the homogenity of clusters with respect to the given classification. This measure is particularly suited if we use the same number of clusters as given classes.

- *Classification accuracy:* Given prototypes, we assign a label $C(\vec{w}^j)$ to every prototype $\vec{w}^j$ which is given by the majority of the labels of data points in the receptive field of $\vec{w}^j$. For a data point $\vec{x}^i$, the label of the winner is $C(\vec{w}^{I(\vec{x}^i)})$,

where $I(\vec{x}^i)$ denotes the index of the winner, as before. The classification accuracy of this function is given by

$$C_2 := \sum_{i=1}^{p} \delta \left(c_i = C(\vec{w}^{I(\vec{x}^i)}) \right) / p$$

This measure can be used for any number of prototypes and classes.

For comparison, we consider the clustering which is obtained from the class-wise Mahalanobis distance; this means, the same number of clusters as given classes is considered, its prototypes are set to the centres of the classes, and the matrix is set to the standard Mahalanobis distance of the respective class, i.e. the inverse correlation matrix of the respective class. Note that the class labels have to be known for this procedure. The model corresponds to a Gaussian mixture model with unimodal classes for known class centers and correlation matrices. We refer to the result by direct Mahalanobis. Note that neither NG nor SOM has been designed for classification purposes, such that the results can only serve as an indicator of the capacity of matrix-enhanced learning. However, most evaluation measures which are based on unsupervised information only (such as e.g. quantization error, trustworthiness, ...) depend on the metric of data points, thus a fair comparison of euclidean and non-euclidean clustering using these alternatives is not possible.

The results (error measure C_1 and C_2 and the standard deviation) of standard NG, SOM, and k-means and NG, SOM, and k-means with matrix learning are given in Tab. 1, whereby we use the same number of cluster centres as given classes. Standard clustering yields isotropic clusters whereas matrix learning adapts ellipsoidal shapes according to the data. Obviously, neighborhood integration improves the robustness and classification ability of the method. Further, matrix adaptation accounts for an improvement of 1-9% depending on the data set due to the larger flexibility of the metric. Interestingly, the result of a direct Mahalanobis distance is worse compared to matrix clustering in breast cancer and ionosphere data sets, although the method uses the given class information and the same number of prototypes as for matrix clustering is available. This can be assigned to the fact that noise in the data set disrupts the relevant information. Unlike the direct Mahalanobis method, matrix clustering uses only those data for matrix computation which lies in the receptive field of the prototype, i.e. the geometric form is taken into account instead of the prior labeling, thus, better robustness with respect to outliers is obtained.

Naturally, the direct Mahalanobis method cannot be used in the case of unknown prior labeling or in the case of multimodal classes which should be represented by more than one prototype. Matrix clustering can directly be used in these settings.

Image compression

Vector quantization on the one hand and principal component analysis on the other hand constitute standard methods for lossy image compression [16, 5]. For vector quantization, images are decomposed into blocks of size $m = m_1 \times m_1$ and a standard vector

coherence C_1	matrix k-Means	matrix SOM	matrix NG	direct Mahalanobis
iris	0.8877 ± 0.0885	0.8917 ± 0.0838	0.9009 ± 0.0778	0.9740
breast cancer	0.8119 ± 0.0553	0.8243 ± 0.0407	0.8445 ± 0.0345	0.8194
ionosphere	0.6076 ± 0.0916	0.5969 ± 0.0647	0.6083 ± 0.0543	0.5323
	k-Means	SOM	NG	
iris	0.8446 ± 0.0521	0.8591 ± 0.0414	0.8737 ± 0.0000	
breast cancer	0.7504 ± 0.0000	0.7504 ± 0.0000	0.7504 ± 0.0000	
ionosphere	0.5871 ± 0.0068	0.5882 ± 0.0011	0.5868 ± 0.0008	
accuracy C_2	matrix k-Means	matrix SOM	matrix NG	direct Mahalanobis
iris	0.8606 ± 0.1356	0.8909 ± 0.1175	0.9147 ± 0.0847	0.9800
breast cancer	0.8953 ± 0.0226	0.9024 ± 0.0245	0.9135 ± 0.0192	0.8998
ionosphere	0.7320 ± 0.0859	0.7226 ± 0.0673	0.7197 ± 0.0600	0.6410
	k-Means	SOM	NG	
iris	0.8499 ± 0.0864	0.8385 ± 0.0945	0.8867 ± 0.0000	
breast cancer	0.8541 ± 0.0000	0.8541 ± 0.0000	0.8541 ± 0.0000	
ionosphere	0.7074 ± 0.0159	0.7114 ± 0.0013	0.7097 ± 0.0009	

Table 1: Classification results of the clustering methods with and without matrix adaptation for various data sets from UCI. The mean heterogeneity of clusters measured by C and the standard deviation are reported.

quantization using p prototypes in $\mathbb{R}^m$ is applied to this data. Afterwards, every block of the image is substituted by the coordinates of the winning prototype, i.e. every block can be encoded using only $\log_2(p)$ bits as a reference to the corresponding prototype. Since the space to represent the prototype vectors is constant and independent of the size of the image, an average of $\log_2(p)/m$ bits per pixel (bpp) are needed for this compression scheme.

Principal component techniques constitute transformation coding techniques of images, since they represent pixels by means of a parameterized function. Again, images are decomposed into blocks of size $m = m_1 \times m_1$ and a standard PCA is applied to these m-dimensional data vectors. The main k principal components represent the transformation. Every block is then represented by k coefficients (these numbers are stored using T bits, in our case $T = 8$). The image is reconstructed by taking the sum of the k principal component directions weighted by these coefficients instead of the full information. Since the principal components can be stored independently of the image size, this compression method leads to $k \cdot T/m$ bbp.

Naturally, a global principal component can be substituted by local PCA techniques which arise from combined vector quantization and PCA techniques. This way, every block is represented by k PCA coefficients and a reference to the winning prototype.

Reconstruction takes place as the weighted sum of the main principle components attached to the prototype translated by the class center. This way, $(k \cdot T + \log_2(p))/m$ bpp are needed for the compression scheme.

We evaluate matrix clustering for image compression in comparison to a simple PCA scheme with only one set of principle components (referred to as PCA), a standard vector quantization scheme, in this case NG without matrix adaptation (referred to as NG), and to alternative transform coding schemes proposed in [27] and [13]. The method proposed in [13] (referrred to as VQPCA) directly combines vector quantization and PCA in a heuristic way. [27] proposes a probabilistic background for an interleaved PCA and vector quantization by means of a mixture of Gaussians (referred to as MoPPCA). We use a direct Matlab implementation of VQPCA, and we use the code provided in [18] for MoPPCA.

Evaluation of all compression schemes is done by the peak signal-to-noise ratio (PSNR) which is defined as follows: Assume x_i is the value of the original image at a certain position and x_i' the reconstructed value. Assume the image consists of N pixels. The mean squared error (MSE) is defined as $\sum_i \|x_i - x_i'\|^2/N$. The PSNR consists of the scaled MSE

$$\mathrm{PSNR} = 10 \cdot \log_{10}\left(\frac{\mathrm{MAX}^2}{\mathrm{MSE}}\right)$$

where MAX denotes the maximum possible pixel value of the image (e.g. it is 255 for images represented by 8 bits per pixel.) Obviously, the higher the PSNR, the better. We evaluate the PSNR of the left half of the image which is used for training, and, in addition, we report the PSNR of the right half of the image which is solely used for testing to judge the generalization ability of the models. Both values are reported together with the standard deviation.

The results for three different images (Lena, House, and Church) and different bpp rates can be found in Tab. 2. We choose parameters of the algorithms such that the bpp rate becomes comparable (4.25, 2.25, and 1.25 bpp, respectively). The parameters are:

- vector quantization based compression using simple neural gas: number of prototypes $n = 19$, block size $m = 1$ (for $4.2479 \approx 4.25$ bpp), $n = 512$, $m = 4$ (for 2.25 bpp), and $n = 32$, $m = 4$ (for 1.25 bpp)
- transformation encoding by simple PCA: block size $m = 64$, number of principal components $k = 34$ (for 4.25 bpp), $m = 64$, $k = 18$ (for 2.25 bpp), $m = 64$, $k = 10$ (for 1.25 bpp)
- transformation encoding by combined vector quantization and PCA: number of prototypes $n = 16$, block size $m = 16$, number of principal components $k = 8$ (for 4.25 bpp), $n = 16$, $m = 16$, $k = 4$ (for 2.25 bpp), $n = 16$, $m = 16$, $k = 2$ (for 1.25 bpp)

Obviously, transformation encoding by local PCA methods is always superior compared to global PCA. When using the same bpp rate, simple VQ encoding yields in

some cases a better PSNR than local PCA methods, which is mostly due to the fact that local PCA methods require a comparably high number of bits for the representation of the PCA coefficient (8 bits each). This could certainly be reduced. Interestingly, however, the PSNR rate for transormation encoding using local PCA methods is in many cases competitive to VQ based encoding or even superior, although the latter uses smaller window sizes and more prototypes for representation. Interestingly, the three methods considered for compression by means of local PCA seem largely comparable. Thereby, the probabilistic model based on mixtures of Gaussians gives slightly worse PSNR, which might be attributed to the different (probabilistic) cost function instead of (some form of) the standard quantization error.

We show the results of the different image compression methods in Figs. 5, 6, 7. While this low bpp rate does not give satisfactory image reconstruction, as expected, the principled characteristics of the compression methods becomes obvious at this extreme setting: vector quantization based image compression leads to 'pixelized' representations, whereas compression based on global PCA yields a blurred impression. Local PCA averages between these two extremes. This seems particularly appropriate for regular textures such as the brick wall or roof of the House image. It seems less appropriate for the representation of parts of the images which are very detailled and which show a large variance such as Lena's face. Here, a compression based on small windows as taken for standard vector quantization gives a better reconstruction.

The three local PCA methods are very similar with respect to the global impression, however, when looking at image details, a few characteristic differences can be observed. In Figs. 8, 9, a detail of the Lena image and House image, respectively, is shown for a 1.25 bpp compression. Again, one can clearly observe the pixelized compression of standard vector quantization and the blurring of global PCA. Compared to MoPPCA and VQPCA, compression by local PCA using matrix NG seems particularly suited to preserve edges in the images, as one can see from Lena's hat and the door of the house, respectively. Overall, compression by matrix NG seems particularly suited for images which display pronounced textures and edges.

5 Conclusions

We presented an extension of neural based clustering such as NG and SOM towards a full matrix. This way, the metric can take scaling and correlations of dimensions into account. Based on batch optimization, an adaptation scheme for matrix learning has been proposed and convergence of the method has been proved. As demonstrated in experiments, matrix NG can improve the clustering performance compared to versions which rely on isotropic cluster shapes. The method has been demonstrated on several benchmark examples.

The found results support work to derive local PCA methods in previous approaches such as [21]. Unlike these previous methods, the presented approach relies on a single cost function and convergence is guaranteed. As demonstrated in the experiments, the convergence behavior of the method is very good in practical applications. Note that,

	4.25 bpp	2.25 bpp	1.25 bpp
Lena			
NG (train)	39.4967±0.0786	37.0219±0.0297	30.0105±0.0382
(test)	36.0334±0.3119	34.6203±0.0248	29.8789±0.2107
PCA (train)	37.4612±0.0000	31.5699±0.0000	28.4870±0.0000
(test)	38.1723±0.0000	32.8029±0.0000	29.8617±0.0000
VQPCA (train)	39.6026±0.3241	33.3710±0.1680	29.7557±0.0830
(test)	40.2752±0.4355	34.2123±0.2086	29.9780±0.1115
MoPPCA (train)	38.2034±0.1278	32.6634±0.1248	29.6081±0.0608
(test)	39.6010±0.4374	33.5733±0.2238	30.1682±0.2223
matrix NG (train)	39.3886±0.1177	32.8789±0.0408	29.3469±0.0512
(test)	40.3762±0.2122	34.2653±0.1468	30.4726±0.1246
House			
NG (train)	41.6597±0.3472	41.4012±0.1252	33.9519±0.1021
(test)	36.8280±0.6140	35.8252±0.1290	30.5782±0.1601
PCA (train)	43.9537±0.0000	37.3119±0.0000	33.3293±0.0000
(test)	40.5958±0.0000	34.2860±0.0000	30.4564±0.0000
VQPCA (train)	45.6874±0.3366	39.3378±0.1088	35.3971±0.1180
(test)	41.9069±0.2468	35.8355±0.0957	31.7602±0.2207
MoPPCA (train)	44.3200±0.3305	38.5901±0.2378	34.7363±0.1742
(test)	41.1701±0.1731	35.2210±0.4565	31.330±0.1499
matrix NG (train)	45.2712±0.1515	39.0032±0.0926	34.5012±0.1394
(test)	42.0525±0.2625	35.9050±0.1391	31.5334±0.0471
Church			
NG (train)	35.3130±0.4311	31.0077±0.1002	25.4105±0.0419
(test)	35.5276±0.2295	28.3431±0.0631	23.4840±0.0316
PCA (train)	29.9463±0.0000	25.4431±0.0000	23.1356±0.0000
(test)	25.8948±0.0000	22.2513±0.0000	20.7304±0.0000
VQPCA (train)	32.6274±0.0950	27.4652±0.0411	24.6618±0.1747
(test)	28.1356±0.0602	23.8068±0.0594	21.7557±0.1484
MoPPCA (train)	31.8378±0.1382	27.1528±0.0441	24.3518±0.0575
(test)	27.5666±0.1279	23.5823±0.0325	21.6546±0.0273
matrix NG (train)	32.6134±0.0347	27.5124±0.0208	24.2528±0.0837
(test)	28.2460±0.0141	23.9505±0.0299	21.8169±0.0088

Table 2: Results of image compression using different compression methods and parameters (as reported in the text) for three different images. The mean PSNR and its standard deviation are depicted.

once the connection of local PCA methods to matrix learning is formally established, the use of heuristics for the optimization steps in batch learning is justified, as long as they improve the respective cost terms. One possibility are online optimization methods which directly optimize the global cost function e.g. by means of a stochastic gradient descent. Alternatively, heuristics such as iterative PCA methods as presented in [21] can be included. These methods seem particularly important if the adaptation of full rank matrices is infeasible due to the data dimensionality. In this case, a restriction to a low rank adaptive approximation can be used. This can be formally justified by means of the same cost function as matrix NG if the eigenvalues in the remaining directions are small.

6 Appendix

We consider the alternative metric

$$d_\Lambda(\vec{x},\vec{w}) = (\vec{x}-\vec{w})\Lambda(\vec{x}-\vec{w})^t + \frac{1}{\lambda^*}((\vec{x}-\vec{w})^t(\vec{x}-\vec{w}) - (\vec{x}-\vec{w})^t\Omega\Omega^t(\vec{x}-\vec{w}))$$

where $\Lambda = \Omega D \Omega^t$ is the singular value decomposition of the rank k matrix Λ given by the k major principal components of the correlation matrix Ω and the inverse of the corresponding eigenvalues D, and λ^* is an estimation of the variance in the remaining $m-k$ directions. In the approach [21], this metric has been integrated into an interleaved PCA and NG scheme to obtain very promising results. We provide a theoretical justification for this procedure under the assumption that the average λ^* is small in comparison to the explicit eigenvalues in Λ.

Assume, as before, the cost function (12)

$$E_{\mathrm{NG}}(\vec{w},\Lambda) \sim \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{p} h_\sigma(k_i(\vec{x}^j)) \cdot d_{\Lambda_i}(\vec{x}^j,\vec{w}^i)$$

where $d_{\Lambda_i}(\vec{x},\vec{w}^i)$ is determined by a low rank matrix with rank k, i.e. we reduce the degrees of freedom of Λ_i to symmetric positive definite matrices with determinant 1 which can be decomposed into k orthonormal directions with arbitrary scaling and $m-k$ directions with uniform scaling. In formulas, this means that the equation

$$\Lambda_i = \sum_{j=1}^{k} \alpha_j^i \cdot \vec{y}_i^j (\vec{y}_i^j)^t + \sum_{j=k+1}^{m} (\alpha^*)^i \cdot \vec{y}_i^j (\vec{y}_i^j)^t \tag{24}$$

holds with α_j^i, $(\alpha^*)^i > 0$ and orthonormal vectors $\vec{y}_i^j$. As before, we can investigate optimum parameters of the cost function for a batch optimization scheme, i.e. optimum assignments given fixed prototypes and matrix parameters, and, in turn, optimum prototypes $\vec{w}^i$, and optimum matrices Λ_i characterized by orthonormal directions $\vec{y}_i^j$ and scaling terms α_j^i and $(\alpha^*)^i$, given fixed assignments. If the respective parameters are

set to the optimum values (or set in such a way that the cost function is decreased in every step), convergence is guaranteed which can be seen in the same way as before.

Obviously, optimum rank assignments are given by the same formula as for standard matrix NGt', the same holds for optimum prototypes. Thus, it remains to compute optimum matrices under the constraints posed on Λ_i. This can be computed independently for every matrix Λ_i, thus we omit the index i for simplicity. Further, for simplicity, we set $\vec{z}^j := \sqrt{h_\sigma(k_{ij})} \cdot (\vec{w}^i - \vec{x}^j)$ and $C := \sum_j \vec{z}^j (\vec{z}^j)^t$ as the generalized Mahalanobis distance which includes neighborhood cooperation. We assume that the eigenvalues of C are pairwise different (which is fulfilled almost surely). Then, the potential for matrix $\Lambda = \Lambda_i$ has the form

$$E(\Lambda) := \sum_j (\vec{z}^j)^t \Lambda \vec{z}^j = \sum_{l \leq k} \alpha_l \cdot (\vec{y}^l)^t C \vec{y}^l + \sum_{l>k} \alpha^* \cdot (\vec{y}^l)^t C \vec{y}^l \tag{25}$$

which has to be minimized for orthonormal $\vec{y}^l$, and α_l, $\alpha^* > 0$ such that $\prod_l \alpha_l \cdot (\alpha^*)^{m-k} = 1$.

Step 1: The derivative of $E(\Lambda) + \sum_l \lambda_l ((\vec{y}^l)^t \vec{y}^l - 1)$ with Lagrange parameter λ_l enforcing the normality of the vectors yields $2\alpha_l C^t \vec{y}^l + 2\lambda_l \vec{y}^l \stackrel{!}{=} 0$ for $l \leq k$ and $2\alpha^* C^t \vec{y}^l + 2\lambda_l \vec{y}^l \stackrel{!}{=} 0$ for $l > k$, hence all vectors $\vec{y}_l$ are eigenvectors of C.

Step 2: Assume the eigenvalue for eigenvector $\vec{y}^l$ is $\lambda_{\vec{y}^l}$. We consider the derivative of $E(\Lambda) + \lambda \cdot \prod_{l' \leq k} \alpha_{l'} \cdot (\alpha^*)^{m-k}$ with Lagrange parameter λ enforcing determinant one with respect to α_l and α^*. The derivative with respect to α_l for $l \leq k$ yields $(\vec{y}^l)^t C \vec{y}^l + \lambda \prod_{l' \leq k, l' \neq l} \alpha_{l'} \cdot (\alpha^*)^{m-k} \stackrel{!}{=} 0$, hence, setting $K := -\lambda \prod_{l' \leq k} \alpha_{l'} \cdot (\alpha^*)^{m-k}$, we find $\alpha_l = K / \lambda_{\vec{y}^l}$. Similarly, the derivative with respect to α^* yields $\sum_{l'>k} (\vec{y}^{l'})^t C \vec{y}^{l'} + \lambda \prod_{l' \leq k} \alpha_{l'} \cdot (m-k) \cdot (\alpha^*)^{m-k-1} \stackrel{!}{=} 0$ hence $\alpha^* = K / (\sum_{l'>k} \lambda_{\vec{y}^{l'}} / (m-k))$. Because the determinant must yield 1, we find $K = \prod_{l \leq k} \lambda_{\vec{y}^l} \cdot (\sum_{l'>k} \lambda_{\vec{y}^{l'}} / (m-k))^{m-k}$, i.e. up to normalization, the coefficient α_l is given by the inverse eigenvalue of $\vec{y}^l$ for $l \leq k$, and α^* is given by the inverse of the average eigenvalues of $\vec{y}^l$ for $l > k$.

Step 3: We want to show that the optima of $E(\Lambda)$ are obtained if $\vec{y}^l$ for $l \leq k$ refers to the k eigenvectors with largest eigenvalues of C under the assumption that the eigenvalues $\vec{y}^l$ are small for $l > k$. We will derive an explicit bound on the eigenvalues which guarantees this fact. Assume two eigenvectors $\vec{y}^p$ and $\vec{y}^q$ are chosen with $\lambda_{\vec{y}^p} > \lambda_{\vec{y}^q}$. We want to show that $E(\Lambda)$ is smaller if $p \leq k$ and $q > k$ in comparison to the value $E(\Lambda)$ which is obtained for $p > k$ and $q \leq k$ under assumptions on the size of the eigenvalues. By repetition, we can then conclude that the indices $l \leq k$ refer to the k largest eigenvalues of C. If we choose $\vec{y}^l$ as eigenvectors of C and α_l, α^* corresponding to the eigenvalues, $E(\Lambda)$ is proportional to the normalization constant K:

$$E(\Lambda) = m \cdot \prod_{l \leq k} \lambda_{\vec{y}^l} \cdot \left(\frac{\sum_{l>k} \lambda_{\vec{y}^l}}{m-k} \right)^{m-k}$$

This value is smaller if $p \leq k$ and $q > k$ than for the case $p > k$ and $q \leq k$ if and only if $\lambda_{\vec{y}^p} \prod_{l \leq k, l \neq p} \lambda_{\vec{y}^l} \cdot \left(\frac{\sum_{l>k, l \neq q} \lambda_{\vec{y}^l} + \lambda_{\vec{y}^q}}{m-k} \right)^{m-k} \leq \lambda_{\vec{y}^q} \prod_{l \leq k, l \neq q} \lambda_{\vec{y}^l} \cdot \left(\frac{\sum_{l>k, l \neq p} \lambda_{\vec{y}^l} + \lambda_{\vec{y}^p}}{m-k} \right)^{m-k}$.

This is equivalent to the inequality

$$\sum_{l>k, l\neq p,q} \lambda_{\vec{y}^l} \leq \frac{\lambda_{\vec{y}^p}(\lambda_{\vec{y}^q})^{1/(m-k)} - \lambda_{\vec{y}^q}(\lambda_{\vec{y}^p})^{1/(m-k)}}{(\lambda_{\vec{y}^p})^{1/(m-k)} - (\lambda_{\vec{y}^q})^{1/(m-k)}} \tag{26}$$

where we exclude p and q in the sum on the left hand side. The bound on the right side of (26) becomes large for $\lambda_{\vec{y}^p} \gg \lambda_{\vec{y}^q}$. Hence indices of the dominant eigenvectors are contained in the set $l \leq k$ for optimum solutions if the variance in the remaining directions is small enough. For intermediate eigenvectors, the optimum is not clear and depends on the situation at hand – however, it can be expected that the exact order of the intermediate eigenvectors has only a minor effect on the value of the cost function $E(\Lambda)$. In the particularly relevant case that k is chosen as the intrinsic dimensionality of the data cluster, the inequality (26) is usually fulfilled for all indices $p \leq k$ and $q > k$ if and only if the indices $l \leq k$ correspond to the k major eigendirections, since a huge gap is present between the value $\lambda_{\vec{y}^k}$ and $\lambda_{\vec{y}^{k+1}}$, the latter resulting only from noise.

Thus, if the inequality (26) holds for the chosen eigenvalue directions, convergence of the method is guaranteed. In this case, matrix learning can be accelerated: instead of a full matrix inversion, the major k eigenvectors and eigenvalues of the generalized correlation matrix are determined, leading to a low rank matrix $\Lambda = \Omega \cdot D \cdot \Omega^t$. The variation in the remaining directions is given by the term $\sum_j (\vec{z}^j (\vec{z}^j)^t - \Omega^t \vec{z}^j (\Omega^t \vec{z}^j)^t)/(m-k)$ where $\vec{z}^j = \sqrt{h_\sigma(k_{ij})} \cdot (\vec{w}^i - \vec{x}^j)$ as before. After normalization such that the determinant is 1, distances can be directly computed.

References

[1] B. Arnonkijpanich and C. Lursinsap, Adaptive Second Order Self-Organizing Mapping for 2D Pattern Representation, in IEEE-INNS IJCNN 2004, 775-780

[2] Asuncion, A. and Newman, D.J. (2007). UCI Machine Learning Repository [http://www.ics.uci.edu/ mlearn/MLRepository.html]. Irvine, CA: University of California, School of Information and Computer Science.

[3] L. Bottou and Y. Bengio (1995), Convergence properties of the k-means algorithm, in NIPS 1994, 585-592, G. Tesauro, D.S. Touret zky, and T.K. Leen (eds.), MIT.

[4] M. Cottrell, J.C. Fort, and G. Pagès (1999), Theoretical aspects of the SOM algorithm, Neurocomputing 21:119-138.

[5] Cottrell, G., Munro, P., and Zipser, D. (1987). Learning Internal Representations from Grey-Scale Images: An Example of Extensional Programming. In Ninth Annual Conference of the Cognitive Science Society, pages 462-473. Hillsdale Erlbaum.

[6] Cottrell, M., Hammer, B., Hasenfuss, A., and Villmann,T.: Batch and median neural gas, Neural Networks, 19 (2006) 762-771

[7] R.O. Duda, P.E. Hart, and D.G. Storck (2000), Pattern Classification, Wiley.

[8] J.-C. Fort, P. Letrémy, and M. Cottrell (2002), Advantages and drawbacks of the Batch Kohonen algorithm, in ESANN'2002, M. Verleysen (ed.), 223-230, D Facto.

[9] Hammer, B. and Hasenfuss, A.: Relational Neural Gas. In J. Hertzberg et al., editors, KI 2007, 190-204.

[10] J.A.Hartigan (1975), Clustering Algorithms, Wiley.

[11] T. Heskes (2001), Self-organizing maps, vector quantization, and mixture modeling, IEEE Transactions on Neural Networks, 12:1299-1305.

[12] F. Höppner, F. Klawonn, R. Kruse, and T. Runkler: Fuzzy Cluster Analysis. Wiley (1999), Chichester.

[13] Kambhatla, A. and Leen, T.K. (1997), Dimension reduction by local principal component analysis, Neural Computation p:1493-1516.

[14] Kohonen, T.: Self-Organizing Maps. Springer, 1995.

[15] Kohonen, T., Kaski, S., and Lappalainen, H.: Self-organized formation of various invariant-feature filters in the Adaptive-Subspace SOM. Neural Computation 9:6, (1997) 1321-1344

[16] Krishna, K., Ramakrishnan, K.R., and Thathachar, M.A.L. (1997). Vector quantization using genetic K-means algorithm for image compression. In: International Conference on Information, Communications and Signal Processing, ICICS. pp. 1585 - 1587, vol.3.

[17] López-Rubio, E., Muñoz-Pérez, J., and Gómez-Ruiz, J.A.: A principal components analysis self-organizing map. Neural Networks 17:2, (2004) 261-270

[18] L.J.P. van der Maaten, E.O. Postma, H.J. van den Herik (2007). Matlab Toolbox for Dimensionality Reduction. MICC, Maastricht University. [http://www.cs.unimaas.nl/l.vandermaaten/Laurens_van_der_Maaten/Matlab _Toolbox_for_Dimensionality_Reduction.html].

[19] Martinetz, T., Berkovich, S., and Schulten, K.: 'Neural gas' network for vector quantization and its application to time series prediction. IEEE TNN 4:4 (1993) 558-569

[20] T. Martinetz and K.J. Schulten (1994), Topology representing networks, Neural Networks 7:507-522.

[21] Möller,R. and Hoffmann, H.: An Extension of Neural Gas to Local PCA. Neurocomputing 62, (2004) 305-326

[22] Mochihashi, D., Kikui, G., and Kita, K.: Learning an optimaldistance metric in a linguistic vector space. Systems and computers in Japan 37:9 (2006) 12-21

[23] J. Peltonen, A. Klami, and S. Kaski (2004), Improved learning of Riemannian metrics for exploratory analysis, Neural Networks, 17:1087-1100.

[24] F.-M. Schleif, Th. Villmann, and B. Hammer: Prototype based fuzzy classification in clinical proteomics, International Journal of Approximate Reasoning 47:1 (2008), 4-16

[25] Schneider, P., Biehl, M., and Hammer, B.: Relevance matrices in LVQ, ESANN 2007, 37-42

[26] Stones, L.Z., Zhang, Y., and Jian, C.L.: Growing Hierarchical Principal Components Analysis Self-Organizing Map. In: Advances in Neural Networks - ISNN 2006, Springer LNCS, 3971 (2006) 701-706

[27] Tipping, M.E. and Bishop, C.M. (1999). Mixtures of probabilistic principal component analyzers, Neural Computation 11:443-482.

[28] S. Zhong and J. Ghosh (2003), A unified framework for model-based clustering, Journal of Machine Learning Research 4:1001-1037.

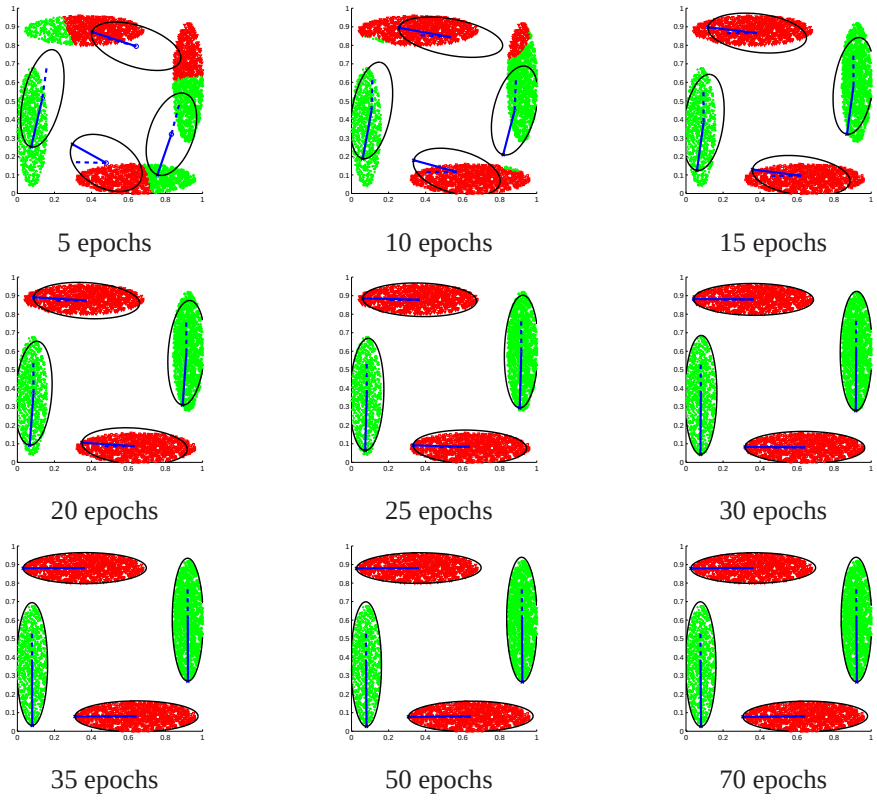

Figure 1: The resulting prototypes and eigenvalues of the matrix for NG and an illustrative two-dimensional data set. The solid line depicts the minor principal component of the found matrix, the dashed line gives the main principal component of the data in the receptive field. Ellipsoids visualize the resulting cluster shapes.

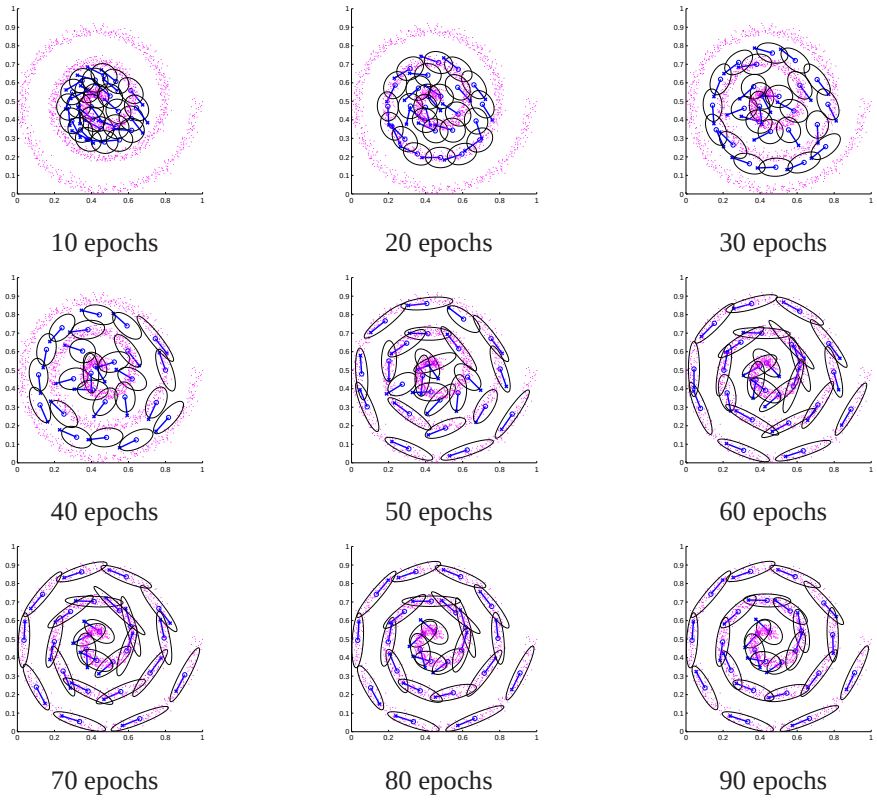

Figure 2: Matrix NG for the spirals data set. Convergence can be observed after about 60 epochs.

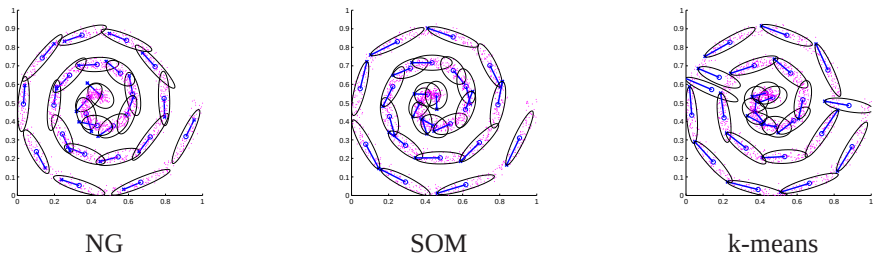

Figure 3: Results of matrix NG, SOM, and k-means after 100 epochs for the spirals data set. Obviously, k-means suffers from local optima.

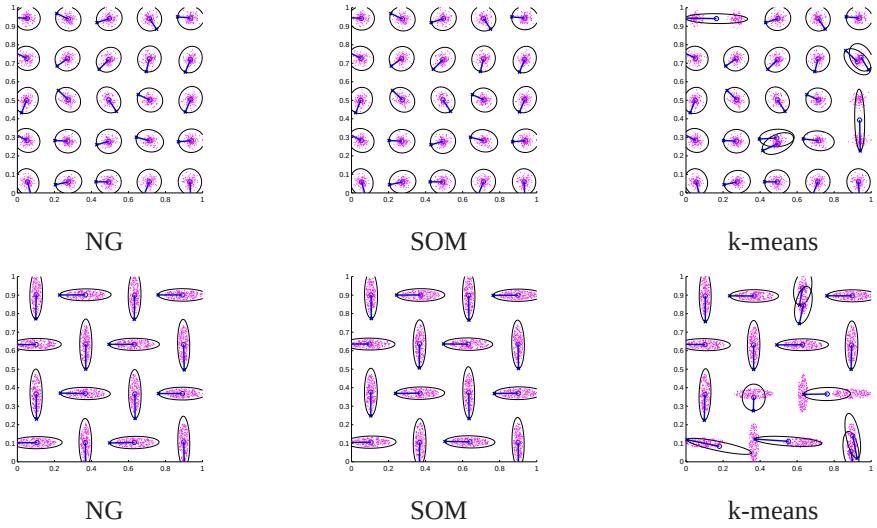

Figure 4: Multimodal checkerboard data with circular shapes (top) and elongated clusters (bottom), respectively. The results of k-means, SOM, and NG after 100 epochs are depicted.

Figure 5: Lena image: original (upper left), and compressed image using 1.25 bpp and neural gas (upper right), global PCA (lower left), and local PCA by means of matrix NG (lower right), respectively.

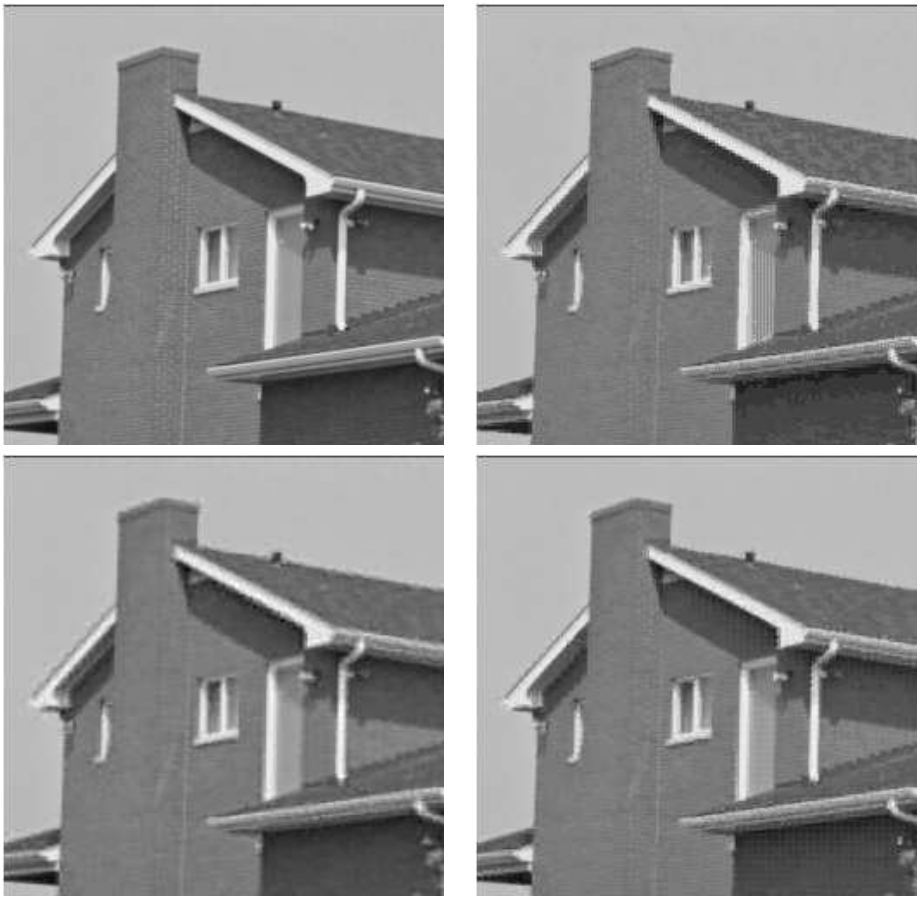

Figure 6: House image: original (upper left), and compressed image using 1.25 bpp and neural gas (upper right), global PCA (lower left), and local PCA by means of matrix NG (lower right), respectively.

Figure 7: Church image: original (upper left), and compressed image using 1.25 bpp and neural gas (upper right), global PCA (lower left), and local PCA by means of matrix NG (lower right), respectively.

Figure 8: Detail of Lena image: original (upper left), and compressed image using 1.25 bpp and neural gas (upper middle), global PCA (upper right), and local PCA by means of VQPCA (lower left), MoPPCA (lower middle) and matrix NG (lower right), respectively.

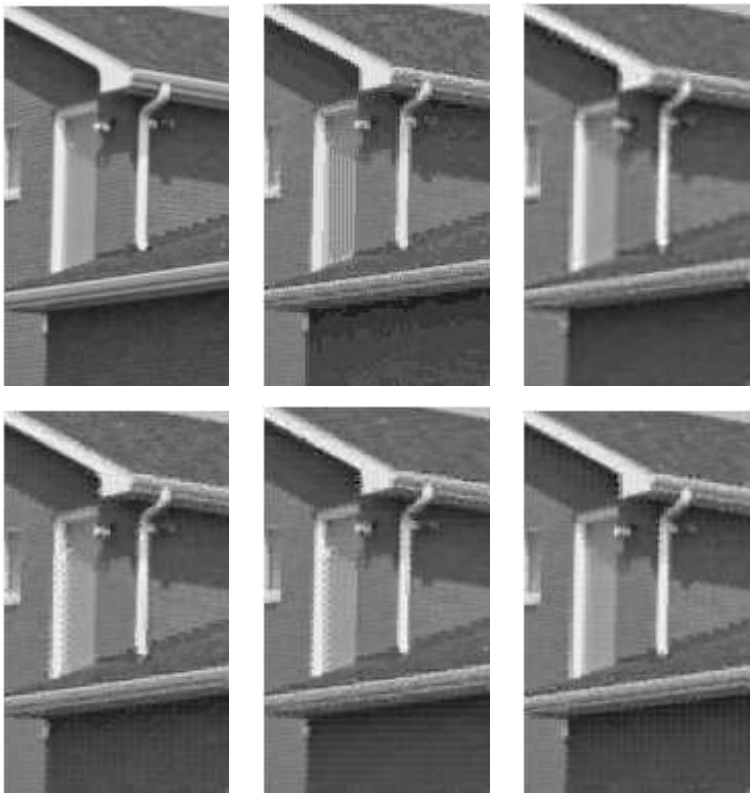

Figure 9: Detail of House image: original (upper left), and compressed image using 1.25 bpp and neural gas (upper middle), global PCA (upper right), and local PCA by means of VQPCA (lower left), MoPPCA (lower middle) and matrix NG (lower right), respectively.